TRAITEMENT D'IMAGES SUR GPU (1)

Algorithmes rapides pour le filtrage et la segmentation des images bruitées sur GPU.

Gilles Perrot

17 avril 2014

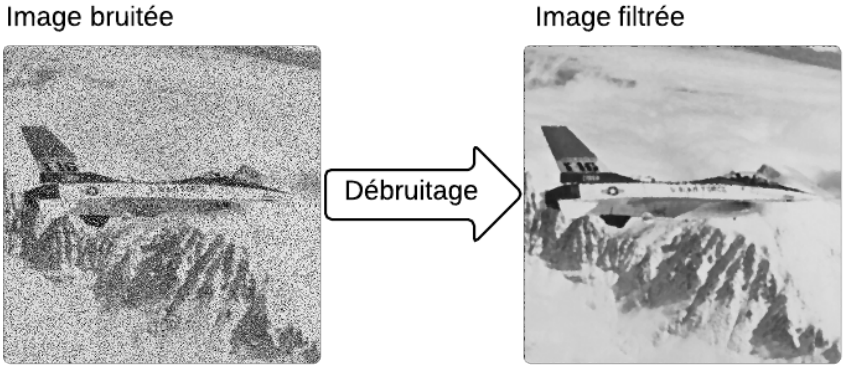
Université de Franche-Comté, Institut FEMTO-ST

Département DISC - équipe AND

Direction : R. Couturier & S. Domas

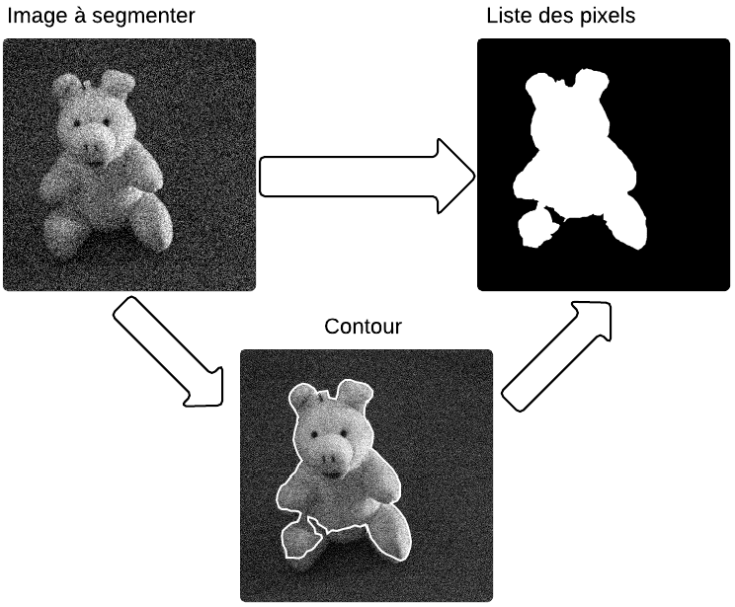
FILTRAGE (2)

Réduire le bruit.



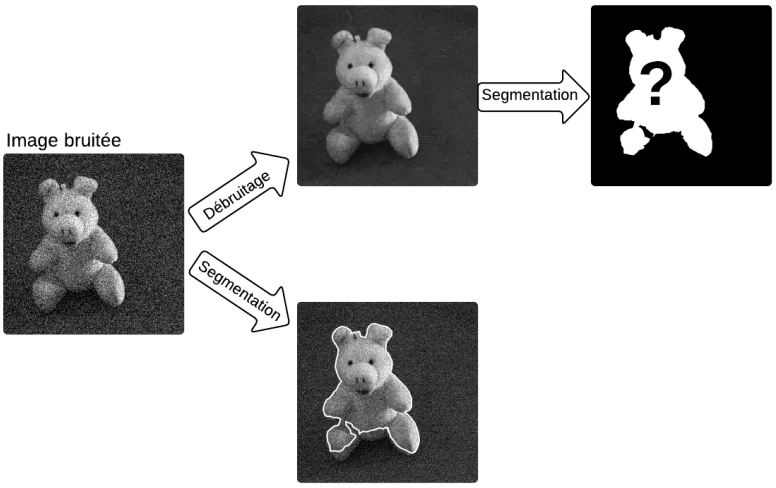
SEGMENTATION (3)

Distinguer les zones homogènes d'une image.



SEGMENTATION (4)

Deux approches

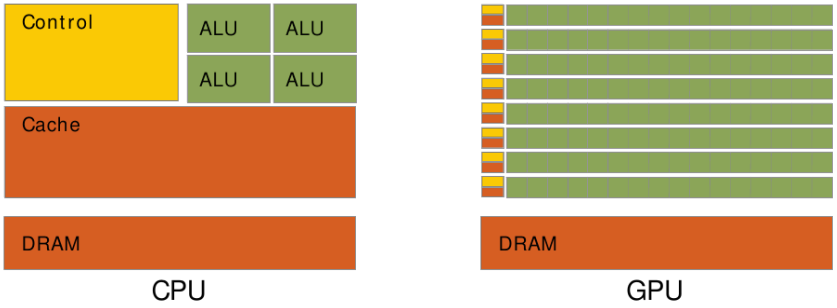


PLAN DE LA PRÉSENTATION (5)

1. Introduction
   - Les GPUs ou *Graphical Processing Units*.
   - Objectifs.
2. La segmentation des images
   - Travaux de référence.
   - Parallélisation GPU d'un algorithme de segmentation de type *snake*.
3. Le filtrage des images
   - Travaux de référence.
   - Optimisation GPU des filtres médian et de convolution.
   - Conception d'un algorithme GPU de débruitage par recherche des lignes de niveaux.
4. Conclusion et perspectives

INTRODUCTION (6)

Les GPUs ou *Processeurs graphiques*.



- Processeurs *classiques* **CPU** : exécution **séquentielle**
  - Quelques unités de calcul (les cœurs).
- Processeurs *graphiques* **GPU** : exécution **massivement parallèle**
  - Des centaines, voire milliers, d'unités de calcul, regroupées en SMs (*Streaming Multiprocessors*).

INTRODUCTION (7)

Ojectif : accélérer

Segmentation

- Algorithme par contours actifs, classe des *snakes*.
- Implémentation CPU optimisée existante.
- Conception de l'implémentation GPU.
- Adaptations mineures de l'algorithme.

Filtrage

- Filtres médians, filtres de convolution
    - Opérateurs mathématiques,
    - Conception d'une implémentation optimisée.
- Filtre par lignes de niveaux
    - Algorithme original,
    - Conceptions conjointes algorithme et implémentation.

SEGMENTATION (8)

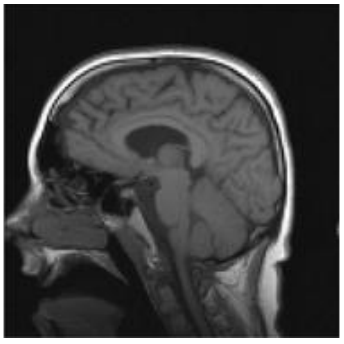
Travaux de référence

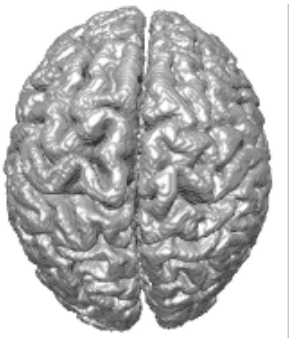
Level-set (Roberts *et al.*, 2010)

- Images d'IRM de 256x256x256 pixels (16 millions),
- Temps sur GTX280 : **11 s.**

Snake GVF (Smistad *et al.*, 2012)

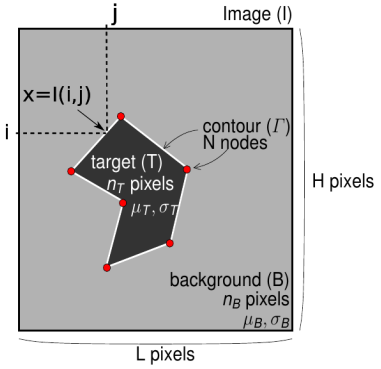
- Images d'IRM de 256x256x256 pixels (16 millions),
- Temps sur C2070 : **7 s.**





SEGMENTATION (9)

Snake polygonal orienté région (principe)



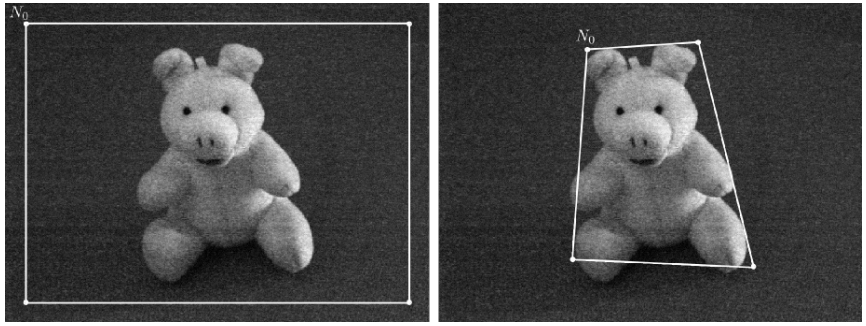
- Objectif : déterminer le contour le plus *vraisemblable*.
- Le critère de vraisemblance généralisée est, dans le cas gaussien :

$$GL = \frac{1}{2}\left[n_B.log(\sigma_B^2) + n_T.log(\sigma_T^2)\right]$$

- Calcul des variances σ^2 pour chaque contour :
  - Méthode de Chesnaud *et al.* (1999) en $\mathcal{O}(n^2)$.
  - Implique le **pré-calcul** de 3 images cumulées.

SEGMENTATION (10)

Snake polygonal orienté région (algo CPU)

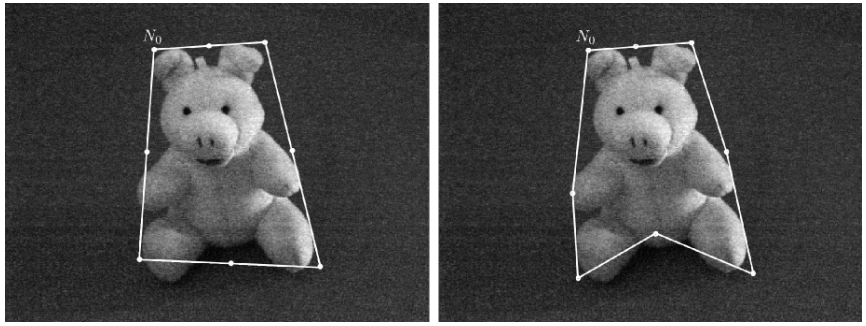


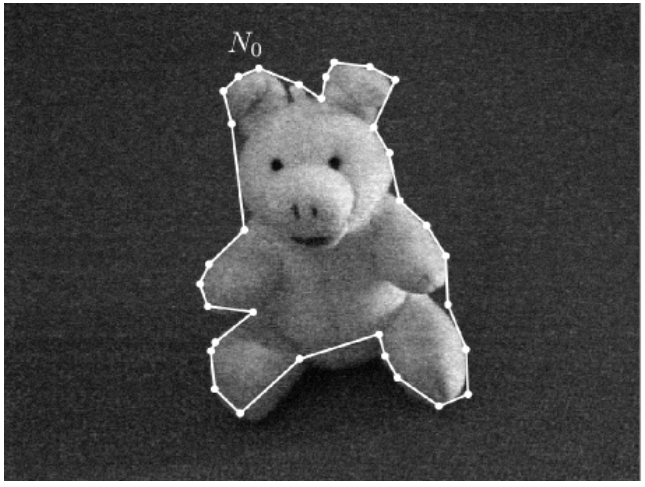
Itération 1

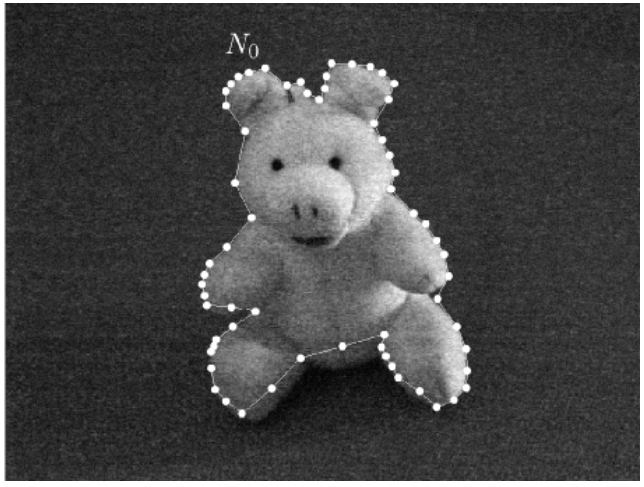
1. Le contour initial est rectangulaire (4 nœuds)
2. On déplace successivement les 4 nœuds jusqu'à ce que plus aucun nouveau déplacement ne provoque l'amélioration du critère.

SEGMENTATION (11)

Snake polygonal orienté région (algo CPU)



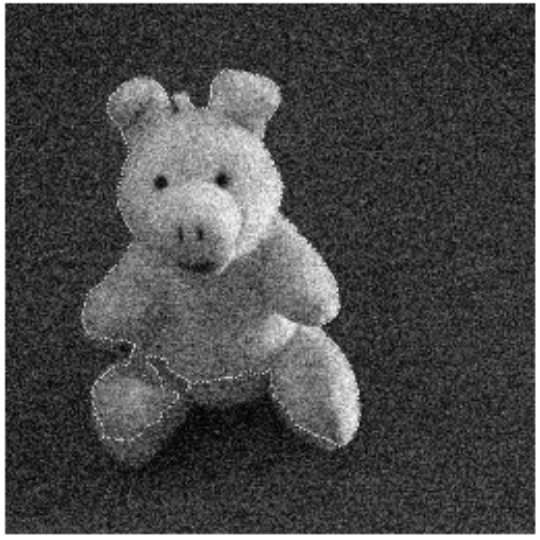




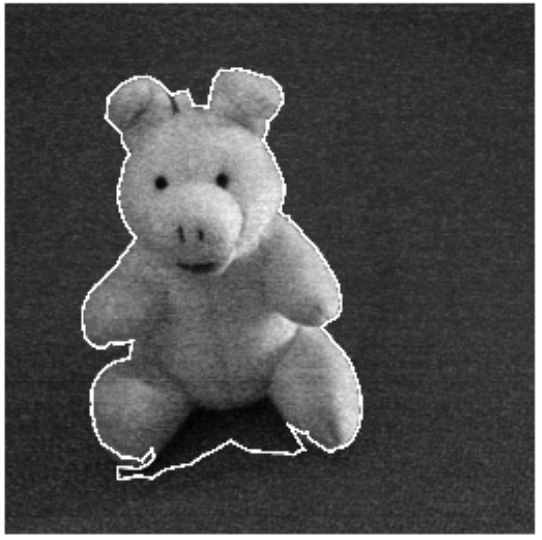
SEGMENTATION (12)

Snake polygonal orienté région (algo CPU)

Exemples de résultats



512x512, 14ms

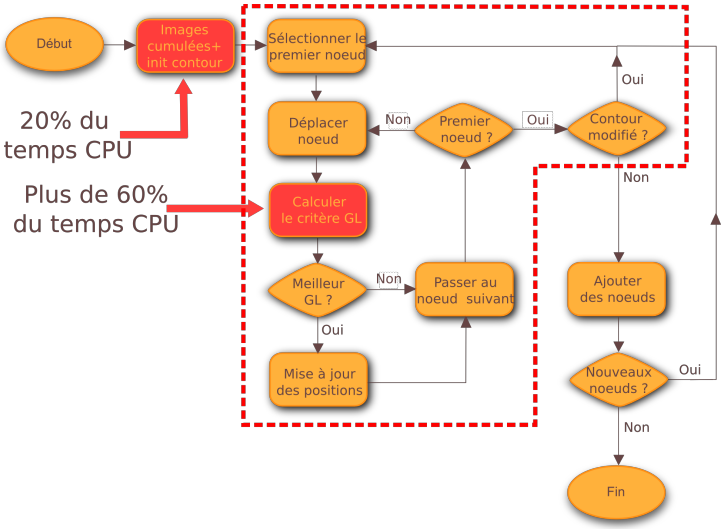


4000x4000, 700ms

SEGMENTATION (13)

Parallélisation du *Snake* polygonal sur GPU

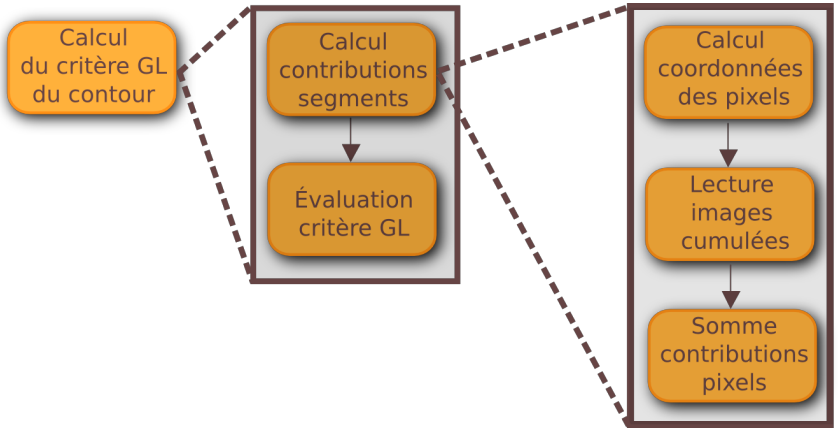
Identification des fonctions coûteuses



SEGMENTATION (14)

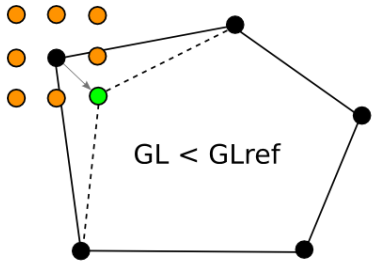
Parallélisation du *Snake* polygonal sur GPU

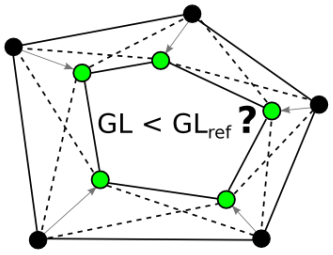
Calcul du critère GL



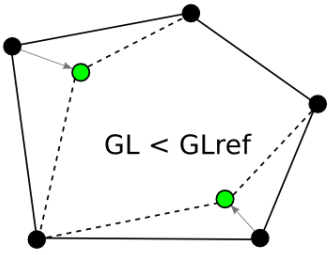
SEGMENTATION (15)

Parallélisation du *Snake* polygonal sur GPU





tous les noeuds
en parallèle



les noeuds pairs/impairs
en parallèle

- position courante
- position envisagée
- position validée

SEGMENTATION (16)

Parallélisation du *Snake* polygonal sur GPU

- 1 thread par pixel.
- Concaténation de tous les pixels composant l'ensemble des contours évalués.
- Réductions en mémoire partagée.
- Une seule taille de segment : la taille du plus long.

SEGMENTATION (17)

Parallélisation du *Snake* polygonal sur GPU

Points positifs :

- Conservation des données en mémoire GPU.
- Images cumulées calculées en parallèle.
- Discretisation des segments en parallèle (1 thread/pixel).
- Respect de l'algorithme original.

Points négatifs :

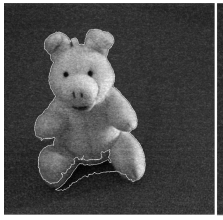
- Trop peu de calculs à effectuer.
- Motifs d'accès à la mémoire globale trop irréguliers.

SEGMENTATION (18)

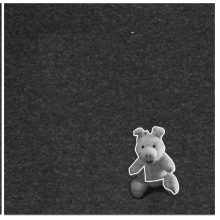
Parallélisation du *Snake* polygonal sur GPU

Performances de l'implémentation

		Performances		
		CPU	GPU	CPU/GPU
	total	**0,51 s**	**0,06 s**	**x8,5**
Image 15 MP	images cumulées	0,13 s	0,02 s	x6,5
(3900×3900)	segmentation	0,46 s	0,04 s	x11,5
	total	**4,08 s**	**0,59 s**	**x6,9**
Image 100 MP	images cumulées	0,91 s	0,13 s	x6,9
(10000×10000)	segmentation	3,17 s	0,46 s	x6,9
	total	**5,70 s**	**0,79 s**	**x7,2**
Image 150 MP	images cumulées	1,40 s	0,20 s	x7,0
(12200×12200)	segmentation	4,30 s	0,59 s	x7,3



100 Mpixels



100 Mpixels

SEGMENTATION (19)

Parallélisation du *Snake* polygonal sur GPU

Conclusion

- Première et seule implémentation connue à ce jour.
- Performances intéressantes pour les grandes images.
- Image 10000x10000 en moins de 0,6 seconde.
- Emploi non optimal du GPU : réductions, irrégularités.
- Premières itérations GPU rapides : grands segments.
- Temps de calcul très dépendant du contenu de l'image.
- Proposition d'une méthode d'initialisation alternative :
    - Recherche du contour rectangle le plus vraisemblable.
    - Accélération jusqu'à x15 avec de petites cibles.

Publication

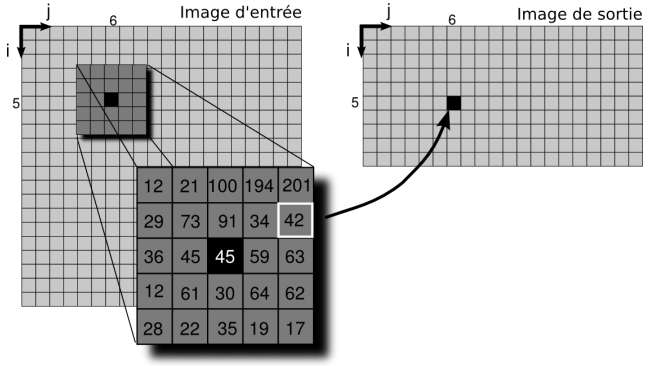
- *G. Perrot, S. Domas, R. Couturier, and N. Bertaux.* **Gpu implementation of a region based algorithm for large images segmentation.** *In Computer and Information Technology (CIT), 2011 IEEE 11th International Conference on, pages 291-298.*

LE FILTRAGE DES IMAGES (20)

- Filtre médian,
- Filtres de convolution,
- Filtre par recherche des lignes de niveaux (PIPD).

LE FILTRAGE DES IMAGES (21)

Filtre médian : principe



La valeur de sortie d'un pixel est la **médiane** des valeurs de son voisinage.

- Bonne réduction de bruits gaussien et *poivre & sel*
- Valeurs de sortie appartenant au voisinage.
- Opération de sélection coûteuse (tri).
- Usages fréquents avec des petites fenêtres (de 3x3 à 7x7).
- Quelques applications avec de grandes fenêtres.

LE FILTRAGE DES IMAGES (22)

Filtre médian : usage

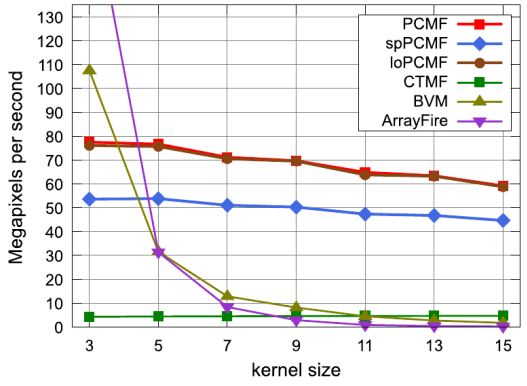
Bruit *poivre & sel* 25% Médian 5x5 Médian 3x3 - 2 passes

LE FILTRAGE DES IMAGES (23)

Filtre médian GPU : Travaux de référence

Comparaison des implémentations GPU de référence :

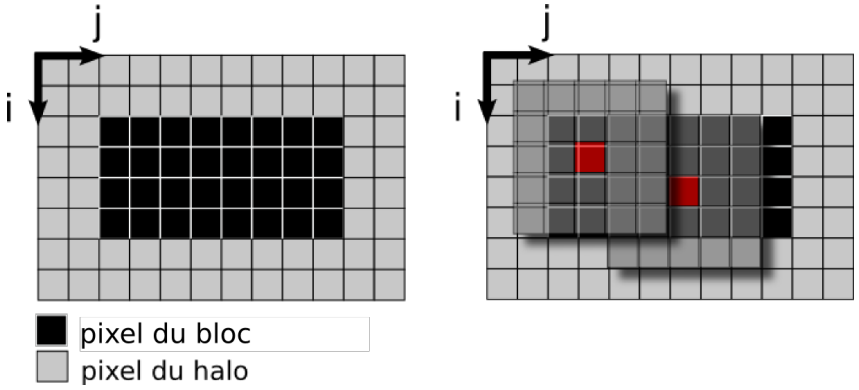
- **PCMF**, Sanchez *et al.* (2013), débit max. **80 MP/s**,
- **ArrayFire**, commerciale (2013), débit max. **185 MP/s**,
- **BVM** parallélisé par Chen *et al.* (2009), débit max. **110 MP/s**.



LE FILTRAGE DES IMAGES (24)

Filtre médian GPU : Travaux de référence

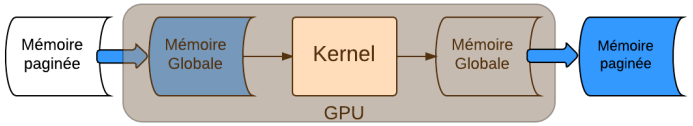
Emploi de la **mémoire partagée** (exemple médian 5x5)



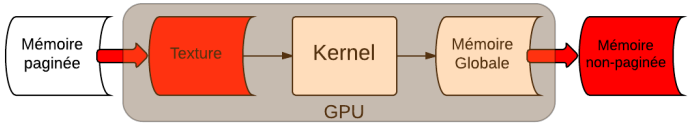
LE FILTRAGE DES IMAGES (25)

Optimisation du filtre médian GPU

Transferts classiques



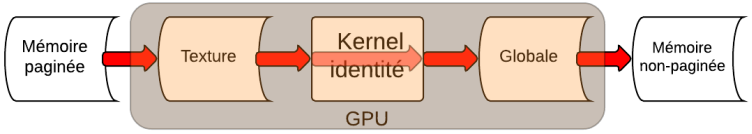
Transferts optimaux



Dimension (pixels)	Profondeur (bits)	**Optimaux** (ms)	**Classiques** (ms)
512×512	8	**0.14**	0.23
	16	**0.24**	0.42
4096×4096	8	**5.88**	7.10
	16	**11.42**	13.16

LE FILTRAGE DES IMAGES (26)

Optimisation du filtre médian GPU



Débits maximums effectifs, en MP/s, sur C2070.

Taille d'image	T_8	T_{16}
512×512	1598	975
4096×4096	2444	1335

Rappel : PCMF : 80 MP/s - BVM : 110 MP/s - ArrayFire : 185 MP/s

LE FILTRAGE DES IMAGES (27)

Optimisation du filtre médian GPU

Sélection de la médiane

- Emploi exclusif des **registres** pour charger les valeurs utiles
    - mémoires individuelles au cœur du GPU,
    - non indexables dans un tableau.
    - maximum de 63 registres par thread et 32 K par bloc de threads.
- Pour les petites tailles : max. 7x7 avec 1 pixel/thread.
- Exploitation des recouvrements entre fenêtres voisines.

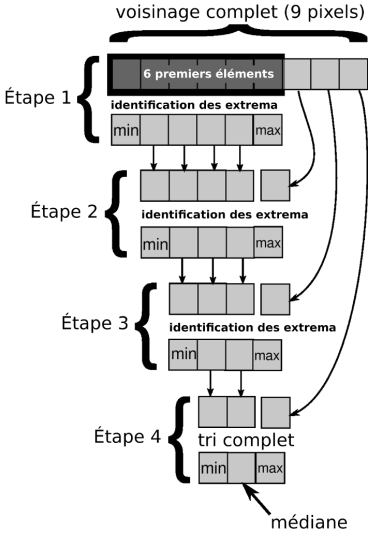
LE FILTRAGE DES IMAGES (28)

Optimisation du filtre médian GPU

Sélection de la médiane (par oubli)

Avantages :

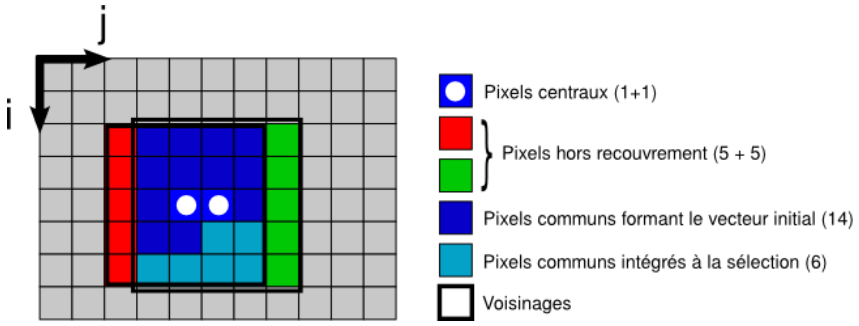
- Évite le tri complet : performances en hausse,
- Économie de registres : $\lceil \frac{n}{2} \rceil + 1$ au lieu de n,
- Permet de plus grandes tailles : max 11x11.



LE FILTRAGE DES IMAGES (29)

Optimisation du filtre médian GPU

Exploitation des recouvrements : 2 pixels par thread (médian 5x5).



À 4 pixels/thread, zone commune = 10 pixels < 14.

LE FILTRAGE DES IMAGES (30)

Performances du médian GPU proposé (PRMF)

Taille d'image	t : temps kernel T_x débit en prof. x	3×3	5×5	7×7
512×512	t (ms)	0.05	0.19	0.60
	T_8 (Mpix/s)	1291	773	348
	T_{16} (Mpix/s)	865	607	307
4096×4096	t (ms)	3.17	11.77	38.06
	T_8 (Mpix/s)	**1854**	951	382
	T_{16} (Mpix/s)	1151	738	340

LE FILTRAGE DES IMAGES (31)

Performances du filtre médian GPU proposé (PRMF)

Image 512x512

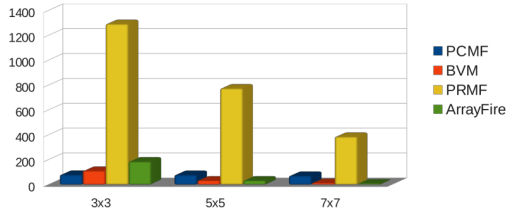
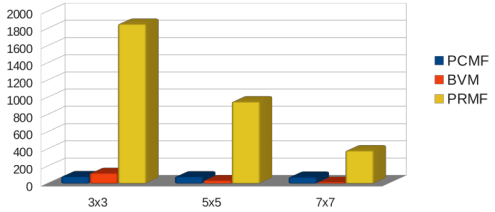


Image 4096x4096



LE FILTRAGE DES IMAGES (32)

Le filtre médian GPU

Conclusion

- Pas d'utilisation de la mémoire partagée.
- Accès optimaux : 1 lecture par pixel.
- Débit global amélioré de **x7** à **x10**, proche du maximum.
- Débit de calcul max. **5,3 GP/s**.
- Médian jusqu'au 11x11 sur C2070, 21x21 sur K40.
- Création d'une application web générant les codes sources.
- Utilisé sur images de cristallographie au synchrotron **SPring-8**.

Publications

- *Gilles Perrot. Image processing. In* **Designing Scientific Applications on GPUs, pages 28,70. CRC Press, 2013.**
- *Gilles Perrot, Stéphane Domas, and Raphaël Couturier.* **Fine-tuned high-speed implementation of a gpu-based median filter.** Journal of Signal Processing Systems, pages 1–6, 2013.*

LE FILTRAGE DES IMAGES (33)

Les filtres de convolution

Principe

$$I_{out}(x,y) = (I_{in} * h) = \sum_{(i<H)} \sum_{(j<L)} I_{in}(x-j, y-i)h(j,i)$$

Selon les valeurs du masque h

- Réduction de bruit, détection de bords,...
- Potentiellement **séparable** en deux convolutions 1D.

LE FILTRAGE DES IMAGES (34)

Les filtres de convolution GPU

Extension des méthodes appliquées au filtre médian :

- Un seul accès mémoire par pixel.
- Mémorisation et calculs en registre.
- Optimum à 8 pixels/thread.

Implémentations de référence (C2070) :

- Nvidia atteint un débit de calcul maximum de **6,00 GP/s**.

LE FILTRAGE DES IMAGES (35)

Les filtres de convolution GPU

Résultats

- Amélioration sensible sur les débits de calcul (de 17 à 33 %).
- Le traitement 1D est jusqu'à 54% plus rapide.
- Débit maximum de **8,54 GP/s**.
- Application à d'autres familles de signaux 1D (audio,...).

LE FILTRAGE DES IMAGES (36)

Filtre par recherche des lignes de niveaux

Motivations :

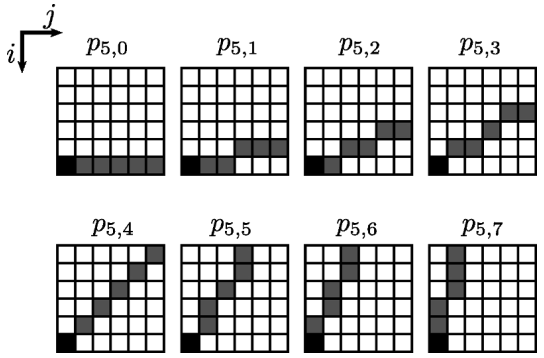
- Les algorithmes qui débruitent le mieux sont lents (BM3D).
- Les images naturelles sont décomposables en un ensemble de lignes de niveaux (iso-niveau de gris).
- Concevoir un algorithme GPU original et son implémentation.

LE FILTRAGE DES IMAGES (37)

Filtre par recherche des lignes de niveaux

Principe - modèle

- Estimation locale, par maximum de vraisemblance.
- Réduction de bruit par moyennage le long de la ligne estimée.
- Les lignes de niveaux estimées sont modélisées par des lignes brisées nommées **isolines** et composées de **segments**.
- Segments choisis parmi 32 motifs pré-calculés.
- Les 8 premiers motifs pour des segments de 6 pixels :



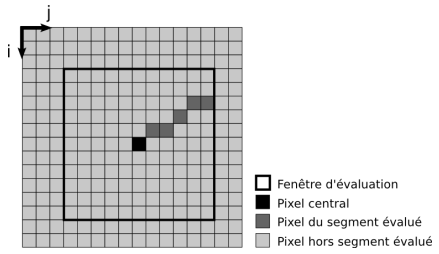
LE FILTRAGE DES IMAGES (38)

Filtre par recherche des lignes de niveaux

Étape 1 (1 pixel/thread)

- En chaque pixel, recherche du motif maximisant la log-vraisemblance (exemple $n = 6$)

$$-\frac{n}{2}\,log(2\pi) - \frac{n}{2}\,log\left(\sigma^2\right) - \frac{n}{2}$$



- Mémorisation des paramètres du motif sélectionné dans deux matrices I_Θ et I_Σ.

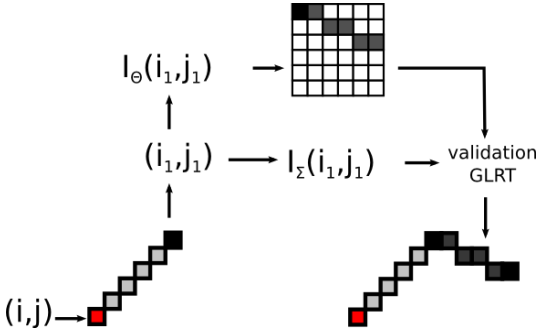
LE FILTRAGE DES IMAGES (39)

Filtre par recherche des lignes de niveaux

Étape 2 (1 pixel/thread)

- Allongement itératif des segments sous condition GLRT.
- Isoline validée de n_{prec} pixels. Candidat de n_s pixels.

$$GLRT = T - (n_{prec}+n_s)\left[log\left(\widehat{\sigma_{prec+s}}^2\right) - log\left(\widehat{\sigma_{prec}}^2\right) - log(\widehat{\sigma_s}^2)\right]$$



LE FILTRAGE DES IMAGES (40)

Filtre par recherche des lignes de niveaux

Étape 3 (optionnelle)

Compensation de la non robustesse de sélection des motifs dans les zones homogènes.

- Conception d'un détecteur de bords.
- Identification des zones homogènes avec ce détecteur.
- Application d'un filtre moyenneur dans les zones identifiées comme homogènes (convolution).

LE FILTRAGE DES IMAGES (41)

Filtre par recherche des lignes de niveaux

Résultats

Ensemble d'images de S. Lansel (DenoiseLab, Université Stanford), filtre proposé (PI-PD)

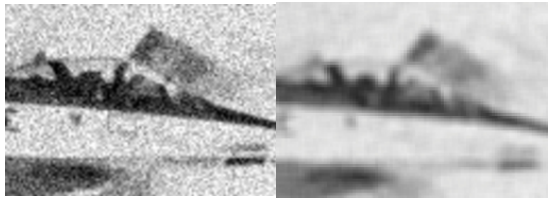
- Amélioration moyenne du rapport Signal sur bruit: **+7,1 dB**
- Indice de similarité structurelle : **+30%**
- Temps de calcul (C2070, **avec** détecteur de bords) : **7 ms**

Algorithme de référence BM3D

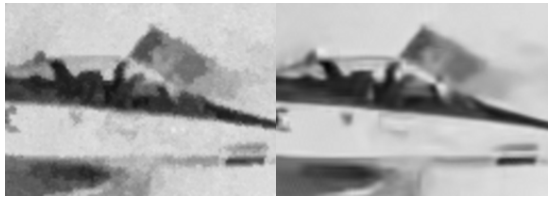
- Amélioration moyenne du rapport Signal sur bruit: **+9,5 dB**
- Indice de similarité structurelle : **+36%**
- Temps de calcul : **4300 ms**

LE FILTRAGE DES IMAGES (42)

Filtre par recherche des lignes de niveaux



Bruit gaussien $\sigma = 25$ Moyennage 5x5



PI-PD BM3D

LE FILTRAGE DES IMAGES (43)

Filtre par recherche des lignes de niveaux

Synthèse - conclusion

- Rapport qualité/temps élevé.
- Traitement d'images HD à 20 fps.
- Artefacts en marche d'escalier.
- Parallélisation de la méthode de Buades *et al.* (2006) :
    - gain +1 dB en +0,2 ms pour 512x512.
- Algorithme original sans implémentation séquentielle de référence.

Publication

- *Gilles Perrot, Stéphane Domas, Raphaël Couturier, and Nicolas Bertaux.* **Fast gpu-based denoising filter using isoline levels.** *Journal of Real-Time Image Processing, pages 1-12, 2013.*

CONCLUSION GÉNÉRALE (44)

- Trois types de conception mis en œuvre.
- Le portage efficace d'algorithme sur GPU peut s'avérer très complexe.
- Certains algorithmes ne se prêtent pas à la parallélisation GPU.
- L'emploi de la mémoire partagée n'apporte pas les meilleures performances en cas de recouvrements.
- Filtrage à des débits inégalés.
- Filtres utilisables par tout programmeur grâce au générateur de code.

PERSPECTIVES (45)

- Extension des filtres aux images couleurs et autres types de bruits (multiplicatif).
- Extension aux pseudo-médians de grandes tailles (microscopie).
- Beaucoup de traitements et domaines peuvent bénéficier des techniques proposées.
- Application aux algorithmes qualitatifs mais lents (BM3D).
- Les évolutions de l'architecture laissent entrevoir de nouvelles possibilités.